The Virtual Interface Architecture Proof-of-Concept Performance Results**

Frank Berry, Ellen Deleganes, Anne Marie Merritt

Server Systems Technology
Intel Corporation
5200 N.E. Elam Young Parkway
Hillsboro, Or. 97124

Abstract

The Virtual Interface Architecture (also called VI Architecture) was developed in response to the need for a new standard communication paradigm for System Area Networks (or SANs). Traditional network communication stacks are designed for use in LANs and WANs and the design tradeoff of generality and distance for performance in these stacks are not optimal for a SAN. Furthermore, the software overhead in these legacy stacks has become more apparent with the advances in hardware network technology. The performance improvements in the hardware are not noticeable at the application level because the software overhead has become the dominant factor in overall network performance.

The design focus for the VI Architecture is to achieve low latency, high bandwidth communication between subsystems in a SAN with minimal CPU usage. The VI Architecture specification is jointly authored by Intel Corporation, Compaq Computer Corporation and Microsoft Corporation. A copy of the specification can be found on the web site at http://www.viarch.org/.

The development of the architecture included the development of prototype VI implementations to validate the architectural concepts and to provide a prototyping platform for application level experimentation. This paper describes the prototyping projects and performance results for each.

Introduction

The VI prototype development was split into two major projects. Both projects resulted in a prototyping platform for application level experimentation, but the development focus for each project was different.

The purpose of the first project was to test the soundness of the overall VI Architecture concepts and to make sure that the architecture could be implemented. The purpose of the second project was to emulate an example hardware implementation and determine what performance gains, if any, could be realized by moving VI functionality from software into hardware.

Architecture Validation Prototype

The VI Architecture validation prototype was built using a standard commodity Ethernet controller with VI functionality emulated entirely in software. The Ethernet hardware selected had no VI specific hardware support or "intelligence" [1]. Thus, the VI functionality had to be emulated completely in software running on the host CPU. Software emulation would have been the approach used in any case because it was the quickest route to validating the VI architectural concepts. Ethernet was selected as the network fabric because it provided a hardware environment that could be easily duplicated for VI application experiments.

Goals

- Full feature implementation to the Rev 0.9 version of the specification.

Rev 0.9 was the latest version of the specification when the prototype project was started.

- Demonstrate at least a two times reduction in overhead as compared to UDP.

An important component of the architecture validation was to demonstrate noticeable

[1] An "intelligent" network interface controller contains an on-board CPU and memory.

performance improvements with VI as compared to legacy protocol stacks. UDP was selected because it has protocol characteristics that are the most similar to VI.

- Provide a stable VI development platform.

The goal was to support at least 4 nodes interconnected in a network, with each node supporting 1000 emulated VIs. Thus, the prototype was not solely a performance demonstration vehicle for VI; it also provided a usable prototyping platform for application level experimentation.

Network Fabric

Ethernet was selected as the network fabric and specifically the Intel Pro100B network interface controllers were used in the development of the prototype. The reasons for selecting this particular hardware configuration follow:

- The controllers and switches are commodity items with hardware and tools, such as network analyzers that are readily available.
- Availability of an NT NDIS driver for UDP for performance comparisons

Performance Measurements and Results

The host nodes used for the performance tests were IA-32 server systems with dual 200 MHZ Pentium ® Pro processors, Intel 82450GX PCIset, and 64MB memory and Intel Pro100B network interface controller.

The software environment for the UDP testing was Microsoft Windows™ NT 3.51 with the Intel supplied Pro100B NDIS driver. The VI performance tests were run on Microsoft Windows™ NT 4.0 with an Intel supplied VI driver.

A modified ttcp test was used to measure the UDP/IP performance. The test was modified only to add timing and round trip information that was not provided by the original test. A test program was written for VI to emulate and measure the same message traffic as the ttcp test. A general description of the latency and bandwidth tests follow.

Application to Application Latency

The Application to Application Latency test measures the time to copy the contents of an application's data buffer to another application's data buffer across an interconnect using the VI Interface. The VI Architecture semantics require a pre-posted receive buffer so the test program includes the time to post a receive buffer.

The test program calculates application to application latency by sending a packet to a remote node that then sends the packet back to the sender. Multiple round trip operations are used to provide an average time per round trip. One half of the average round trip time is the Single Packet Application to Application Latency.

Figure 1 shows the comparison of application to application latency of VI versus UDP. The graphs show that VI has a latency of approximately half of the latency of UDP for the same message sizes.

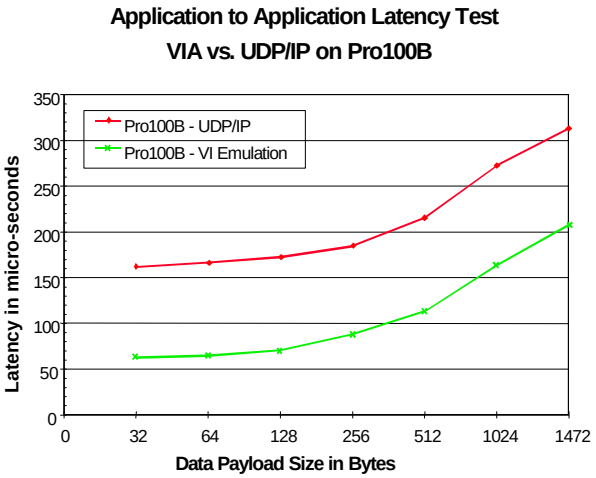

Figure 1: VI vs. UDP Latency

Application Send Bandwidth

The Application Send Bandwidth tests measure the rate at which large amounts of data can be sent from one application to another application across the interconnect. The VI Architecture semantics require pre-posted receive buffers so the test program includes the time to post these receive buffers.

The test program calculates application send bandwidth by initiating multiple concurrent sends then continuing to issue sends as quickly as possible for the duration of the test time. The receiving node must have multiple receives outstanding throughout the test. The bandwidth is calculated simply by dividing the total amount of data sent by the total test time.

The comparison of VI and UDP bandwidth for various message sizes is shown in Figure 2. This graph shows that the reduction in latency for VI also translates to increased bandwidth for all message sizes.

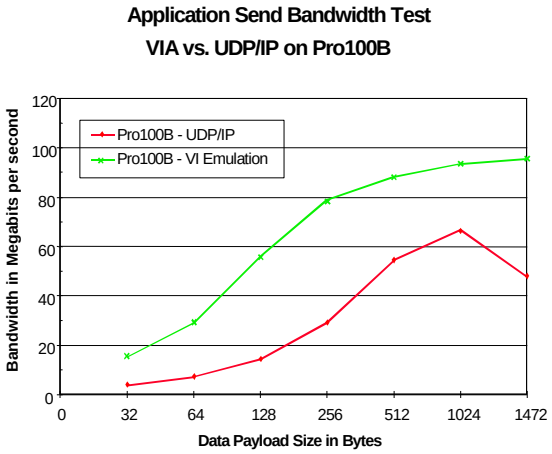

Figure 2: VI vs. UDP Bandwidth

Hardware Emulation Project

The primary focus for this project was to emulate an example hardware design. This change in focus required a change in the hardware selection. The Ethernet hardware used in the initial project could not provide an environment suitable for emulating a hardware design. The hardware selected for this project consisted of "intelligent" network interface controllers and associated switches.

Throughout the following description: A network interface controller coupled with the VI Architecture emulated in software running on the host node is referred to as a VI-emulated NIC. A network interface controller coupled with the VI Architecture implemented in hardware and/or software running on the network interface controller is referred to as a VI-NIC.

Goals

- Validate the VI Architecture concepts and provide a basis for experimentation with applications using the VI Architecture in place of traditional network interfaces.

Prototypes are expected to support at least 4 nodes interconnected with a network, each node supporting at least 200 VIs and mapping of 64 Megabytes of memory per network interface controller.

- Demonstrate performance gains available with a VI-NIC compared to a VI-Emulated NIC.

The performance advantages of a VI-NIC should be clearly visible as compared to emulating the VI interface using a low-cost network interface controller.

- Expose Potential Problems and Issues with the VI-NIC Designs.

Creation of a functional VI-NIC and the software to drive it should allow discovery of any synchronization issues that were undiscovered in the original VI Architecture proof-of-concept.

Network Fabric

Myricom Myrinet network interface controllers and switches were used for the hardware emulation project. The reasons for selecting this hardware are as follows:

- The network fabric supports a data rate of at least one gigabit per second. Specifically, the hardware chosen provides 1.28 Gb/s. This allows a comparison of low-cost, VI-emulated NIC to a native VI-NIC implementation for gigabit class networks.
- The programmable RISC CPU on the network interface controller allows for rapid development and modification of the prototype.
- Enough memory resources are present to allow the control program, memory protection and translation table and per VI context information to reside in on-board memory. Specifically, the network interface controller contains 1 Mbyte of memory.
- The availability of tools and documentation for developing the network interface's control program.
- The availability of example source code for use as a starting point.
- The network interface controllers are "off-the-shelf" products and are compatible with the host's hardware PCI-32 I/O bus and Pentium® Pro Processor.

Prototype Implementations

Two prototype environments were created a VI-Emulated NIC and an emulation of a VI-NIC.

For the VI-Emulated NIC environment, the Myrinet network interface controller was programmed to emulate a low-cost gigabit class network interface controller and the driver software supported the VI emulation. This implementation only provided a means to place packets onto a network and to extract packets from the network. These types of network interface controllers only understand physical addresses and deal with network data on a packet basis, leaving the work of memory address translation and per connection multiplexing and de-multiplexing of the packet stream to the host CPU. This is environment is referred to as "VI Emulated in Kernel Agent" in the performance graphs.

For the emulated VI-NIC environment, the Myrinet network interface controller was programmed to emulate an example VI-NIC

hardware design. The driver software was modified to support this emulation.

This environment is referred to as "VI Emulated on NIC" in the performance graphs.

VI-NIC Characteristics

A VI-NIC works directly with send and receive descriptors, including virtual addresses. The memory translation and protection functions and the multiplexing and de-multiplexing of packets to connections are all handled by the network interface controller. The hardware and software map the doorbell registers directly into the application's memory space to allow the posting of descriptors to avoid a kernel transition.

The VI-NIC emulation that generated the results presented in this paper did not emulate directly accessible doorbell registers, and thus required a kernel transition to post a descriptor.

Performance Measurements and Results

The host nodes used for the performance tests were Micron Brand Millennia® PRO2 systems with 64 MB of DRAM, Dual 200 MHz Pentium® Pro Processors with 256K of L2 cache each, 440 FX chipset and Myrinet network interface controllers.

The software environment for the performance tests was Microsoft Windows™ NT 4.0 with Service Pack 3 and Intel supplied VI drivers.

Application to Application Latency

The Application to Application Latency test is the same test used to gather the VI Ethernet latency numbers described in the Architecture Validation Prototype section of this paper.

The results of the Application to Application Latency test are shown in Figure 3. The latency reduction shown by the VI-NIC is primarily attributable to the elimination of receive side host data copies and host system interrupts. The performance gains achieved were limited primarily by the increased quantity of code executed by the network interface controller's processor.

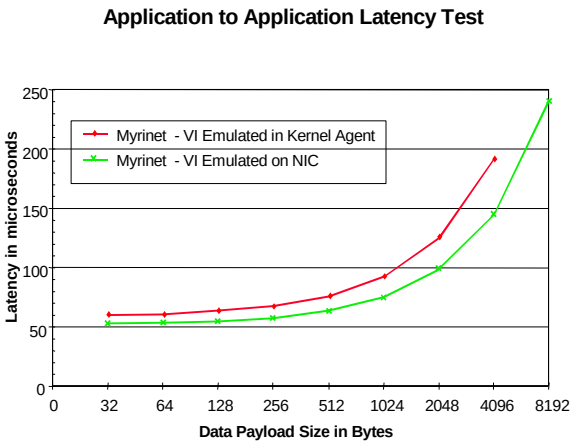

Figure 3: Application to Application Latency

Application to Application Bandwidth

The Application to Application Bandwidth test measures the rate at which large amounts of data can be copied from one application to another application an interconnect using the VI Interface. Since the VI Architecture semantics require pre-posted receive buffers, the test program includes the time to post these receive buffers.

The test program calculates application to application bandwidth by initiating multiple round trip operations (with an average of 50 outstanding round trip operations) and obtaining an average time per round trip. Using one-half of the average round trip time and the data packet payload size, the application to application bandwidth for "streamed" data can be calculated.

The results of the Application to Application Bandwidth test are shown in

Figure 4. The reduction in latency for the VI-NIC translates directly to an increase in bandwidth for all message sizes.

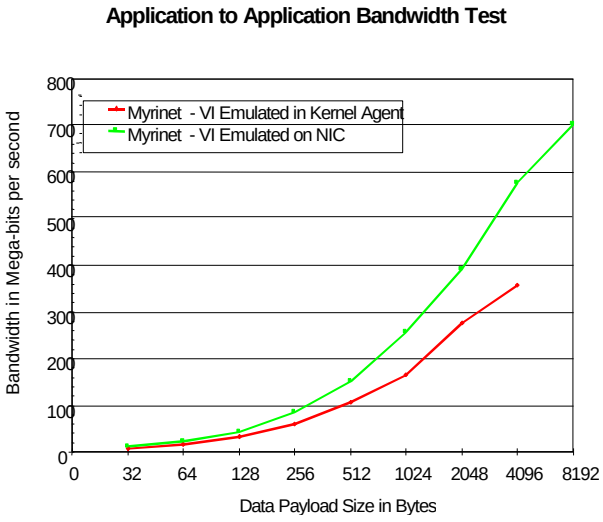

Figure 4: Application to Application Bandwidth

CPU utilization

The Application CPU Load test measures the amount of CPU time that is unavailable to the application due to passing messages. This test program measures the total cost of system activity related to message passing that affects the application. This includes the execution of message passing code (library calls), kernel transitions (if any), interrupt handling, CPU instruction cache flushing (due to execution of instructions outside of the main application) as well as cache snooping and memory bandwidth contention due to movement of data to and from network interface controller.

This test program calculates the overhead to pass a message by constructing a situation where the test application can consume all of the CPU resources while performing its work. This test program uses a vector sum for the work load. The application is run twice, once without passing messages, and once with message passing. Each run is performed for a fixed amount of time and the number of vector sums completed is recorded. The first run yields an average time to perform a vector sum. The second run yields cost of the message passing operations performed during the test run. The second run completes some number of vector sums (always smaller in quantity than the first test run). The time consumed by the message passing work is determined by subtracting the amount of time spent running the application (the number of vector sums performed in the second test run multiplied by the average time to perform a vector sum) from the first test run from the total test time. The system overhead per message is simply the total message passing time divided by the number of messages passed.

The results of the CPU Utilization test are shown in Figure 5. In this test, the entire overhead of handling host system interrupts and performing host system data copies is clearly visible. In the VI-NIC results, the slight increase in overhead associated with increasing the packet size is attributable to the PCI data transfers competing with the application for host system memory bandwidth

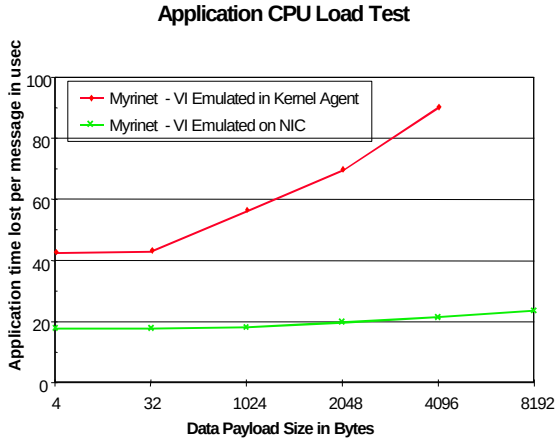

Figure 5: CPU Utilization

Performance Issues

Although the selected hardware's capabilities were well matched to the requirements, a small number of limitations constrained the performance achievable by emulating the VI-NIC hardware.

- The 33 MHz instruction rate of the on-board controller is overshadowed by the host system's 200 MHz Pentium Pro® Processor.
- Due to the Myrinet network interface controller architecture, the transfer of a block of data takes four data copy operations:

  1- Local host to local network interface controller

  2- Local network interface controller memory to network*

  3- Network to remote network interface controller memory*

  4- Local network interface controller memory to host

  * The data copy labeled #2 is mostly overlapped with data copy #3.

Emulation of the VI Doorbell scheme is difficult for more than a trivial number of VIs without consuming a significant amount of memory, controller CPU cycles or PCI bus resources.

Design Limitations

To support the stated project goals, the design focused on emulating an example hardware design and providing a useful application prototyping platform over optimizing performance for the Myrinet network interface controller. If for example, the goal was to design and implement the fastest VI-NIC system for the Myrinet network interface controllers, different design choices may have yielded better performance.

Emulation Overheads

The scheme used to emulate the doorbell functionality required an application to kernel transition. In the NT environment used for test and development, doorbell emulation cost approximately 4.5 us.

Doorbell support in hardware would reduce measured single packet latencies by an estimated 7.5 us and CPU load by and estimated 9 us.

Improving Emulation Performance

Two additions to an intelligent network interface controller would provide significant performance improvements over the prototype results achieved in this project:

- A faster programmable controller would result in a more desirable bandwidth performance curve (higher bandwidth is achieved at smaller message sizes) and would improve small packet latency. Control CPUs in the 200 MIP range are available to hardware implementers today.
- A hardware assist for implementing the doorbell registers would reduce small packet latency and significantly reduce host CPU cost per message sent. FPGA technology is available today that could provide this assist.

Projected Native Hardware VI-NIC Performance

Projected VI-NIC performance numbers have been generated through hardware simulation both within Intel and by external vendors who are planning to develop hardware VI-NICs. The projections indicate that latency for small messages with a native hardware VI-NIC will be in the 10 microsecond range and bandwidth for large packets will be limited by the bandwidth of the interconnect fabric or PCI, whichever is lower.

Conclusion

The results of the first prototype project demonstrate the VI Architecture met its goals to reduce the latency and increase the bandwidth as compared to legacy network stacks. Although performance was not the primary focus, the prototype achieved approximately a 2x reduction in message latency as compared to the UDP/IP communication stack on the same hardware platform.

To emulate an example hardware design, the second prototype project implemented the VI Architecture on a commercially available network interface controller that could be programmed to resemble a hardware VI-NIC. The performance of the hardware VI-NIC emulation was compared against the performance of the host software implementation.

The results show that emulating VI functions in host software is feasible, but that a significant amount of performance is lost in both the communications and application space.

Emulating VI-NIC functionality using intelligent network interface controllers has performance benefits over an implementation that emulates VI functions in host software but cannot match the projected performance of a hardware VI-NIC implementation.

The best network performance and the best price/performance will be available on network interface controllers that implement the core VI functionality in special purpose silicon and eliminates all possible processing overheads in the send and receive paths.

** THIS TEST REPORT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance